The Roots of LISP

A paper by Paul Graham reconstructing McCarty’s original LISP interpreter presented by Marco Valtorta

November 28, 2001
 Seven Primitive Operations

1. Cons
2. & Cond
3. Eq
4. Car (Head of)
5. cdr (Tail)
6. Cons
7. Car (First)
8. cdr (Rest)
9. Car (First)
10. cdr (Rest)
11. Car (First)
12. cdr (Rest)
13. Car (First)
14. cdr (Rest)
15. Car (First)
16. cdr (Rest)
17. Car (First)
18. cdr (Rest)
19. Car (First)
20. cdr (Rest)
21. Car (First)
22. cdr (Rest)
23. Car (First)
24. cdr (Rest)
25. Car (First)
26. cdr (Rest)
27. Car (First)
28. cdr (Rest)
29. Car (First)
30. cdr (Rest)
31. Car (First)
32. cdr (Rest)
33. Car (First)
34. cdr (Rest)
35. Car (First)
36. cdr (Rest)
37. Car (First)
38. cdr (Rest)
39. Car (First)
40. cdr (Rest)
41. Car (First)
42. cdr (Rest)
43. Car (First)
44. cdr (Rest)
45. Car (First)
46. cdr (Rest)
47. Car (First)
48. cdr (Rest)
49. Car (First)
50. cdr (Rest)
51. Car (First)
52. cdr (Rest)
53. Car (First)
54. cdr (Rest)
55. Car (First)
56. cdr (Rest)
57. Car (First)
58. cdr (Rest)
59. Car (First)
60. cdr (Rest)
61. Car (First)
62. cdr (Rest)
63. Car (First)
64. cdr (Rest)
65. Car (First)
66. cdr (Rest)
67. Car (First)
68. cdr (Rest)
69. Car (First)
70. cdr (Rest)
71. Car (First)
72. cdr (Rest)
73. Car (First)
74. cdr (Rest)
75. Car (First)
76. cdr (Rest)
77. Car (First)
78. cdr (Rest)
79. Car (First)
80. cdr (Rest)
81. Car (First)
82. cdr (Rest)
83. Car (First)
84. cdr (Rest)
85. Car (First)
86. cdr (Rest)
87. Car (First)
88. cdr (Rest)
89. Car (First)
90. cdr (Rest)
91. Car (First)
92. cdr (Rest)
93. Car (First)
94. cdr (Rest)
95. Car (First)
96. cdr (Rest)
97. Car (First)
98. cdr (Rest)
99. Car (First)
100. cdr (Rest)
101. Car (First)
102. cdr (Rest)
103. Car (First)
104. cdr (Rest)
105. Car (First)
106. cdr (Rest)
107. Car (First)
108. cdr (Rest)
109. Car (First)
110. cdr (Rest)
111. Car (First)
112. cdr (Rest)
113. Car (First)
114. cdr (Rest)
115. Car (First)
116. cdr (Rest)
117. Car (First)
118. cdr (Rest)
119. Car (First)
120. cdr (Rest)
121. Car (First)
122. cdr (Rest)
123. Car (First)
124. cdr (Rest)
125. Car (First)
126. cdr (Rest)
127. Car (First)
128. cdr (Rest)
129. Car (First)
130. cdr (Rest)
131. Car (First)
132. cdr (Rest)
133. Car (First)
134. cdr (Rest)
135. Car (First)
136. cdr (Rest)
137. Car (First)
138. cdr (Rest)
139. Car (First)
140. cdr (Rest)
141. Car (First)
142. cdr (Rest)
143. Car (First)
144. cdr (Rest)
145. Car (First)
146. cdr (Rest)
147. Car (First)
148. cdr (Rest)
149. Car (First)
150. cdr (Rest)
151. Car (First)
152. cdr (Rest)
153. Car (First)
154. cdr (Rest)
155. Car (First)
156. cdr (Rest)
157. Car (First)
158. cdr (Rest)
159. Car (First)
160. cdr (Rest)
161. Car (First)
162. cdr (Rest)
163. Car (First)
164. cdr (Rest)
165. Car (First)
166. cdr (Rest)
167. Car (First)
168. cdr (Rest)
169. Car (First)
170. cdr (Rest)
171. Car (First)
172. cdr (Rest)
173. Car (First)
174. cdr (Rest)
175. Car (First)
176. cdr (Rest)
177. Car (First)
178. cdr (Rest)
179. Car (First)
180. cdr (Rest)
181. Car (First)
182. cdr (Rest)
183. Car (First)
184. cdr (Rest)
185. Car (First)
186. cdr (Rest)
187. Car (First)
188. cdr (Rest)
189. Car (First)
190. cdr (Rest)
191. Car (First)
192. cdr (Rest)
193. Car (First)
194. cdr (Rest)
195. Car (First)
196. cdr (Rest)
197. Car (First)
198. cdr (Rest)
199. Car (First)
200. cdr (Rest)
201. Car (First)
202. cdr (Rest)
203. Car (First)
204. cdr (Rest)
205. Car (First)
206. cdr (Rest)
207. Car (First)
208. cdr (Rest)
209. Car (First)
210. cdr (Rest)
211. Car (First)
212. cdr (Rest)
213. Car (First)
214. cdr (Rest)
215. Car (First)
216. cdr (Rest)
217. Car (First)
218. cdr (Rest)
219. Car (First)
220. cdr (Rest)
221. Car (First)
222. cdr (Rest)
223. Car (First)
224. cdr (Rest)
225. Car (First)
226. cdr (Rest)
227. Car (First)
228. cdr (Rest)
229. Car (First)
230. cdr (Rest)
231. Car (First)
232. cdr (Rest)
233. Car (First)
234. cdr (Rest)
235. Car (First)
236. cdr (Rest)
237. Car (First)
238. cdr (Rest)
239. Car (First)
240. cdr (Rest)
241. Car (First)
242. cdr (Rest)
243. Car (First)
244. cdr (Rest)
245. Car (First)
246. cdr (Rest)
247. Car (First)
248. cdr (Rest)
249. Car (First)
250. cdr (Rest)
251. Car (First)
252. cdr (Rest)
253. Car (First)
254. cdr (Rest)
255. Car (First)
256. cdr (Rest)
257. Car (First)
258. cdr (Rest)
259. Car (First)
260. cdr (Rest)
261. Car (First)
262. cdr (Rest)
263. Car (First)
264. cdr (Rest)
265. Car (First)
266. cdr (Rest)
267. Car (First)
268. cdr (Rest)
269. Car (First)
270. cdr (Rest)
271. Car (First)
272. cdr (Rest)
273. Car (First)
274. cdr (Rest)
275. Car (First)
276. cdr (Rest)
277. Car (First)
278. cdr (Rest)
279. Car (First)
280. cdr (Rest)
281. Car (First)
282. cdr (Rest)
283. Car (First)
284. cdr (Rest)
285. Car (First)
286. cdr (Rest)
287. Car (First)
288. cdr (Rest)
289. Car (First)
290. cdr (Rest)
291. Car (First)
292. cdr (Rest)
293. Car (First)
294. cdr (Rest)
295. Car (First)
296. cdr (Rest)
297. Car (First)
298. cdr (Rest)
299. Car (First)
300. cdr (Rest)
A Notation For Functions

\[
\text{defn } f \ (p_1 \ldots p_n \ e) \\
\text{let } f \in \text{lambda} (p_1 \ldots p_n \ e)
\]

a function with itself. curiously, because there is no way to name adequate for naming functions defined re.

\[
\text{let notation } \text{The lambda notation is in-}
\]

\[
\text{=} \\
\text{lambda} (p_1 \ldots p_n \ e) \ \text{at} \ an
\]

the desired functions \[\text{McCarthy, p. 186}\].

ring in the form and the \ldots arguments of

\[
\text{response between the variables occur-}
\]

\[
\text{into a function if we can determine the cor-}
\]

\[
\text{ lambda notation } \text{A form can be converted}
\]

A Notation For Functions
\textbf{Study 4 vs (Car 2)}...

\textbf{ft} (\textbf{count} (\textbf{subject} x y (\textbf{Car} 2))

(\textbf{ft} 2)

\textbf{count} (\textbf{expr} 2)

\textbf{false} \textbf{subject} (x y z)

\textbf{subject} x y (\textbf{car} 2))

(\textbf{ft} (\textbf{count} (\textbf{subject} x y (\textbf{car} 2))

(\textbf{ft} 2)

\textbf{count} (\textbf{expr} 2)

\textbf{label} \textbf{subject} (\textbf{lambda} (x y z)

(a m (a m c (b (a b (c d))))

\textbf{subject} m 1 b (a b (c d))
Some Functions

| Lambda | (λ x → (λ y → (if x then y else z))) (null x (foo bar)) | to apply | null | (λ x → (λ y → (if x then y else z))) (null x (foo bar)) |
Some Functions, ct. dv
(car (cdr (assoc x y)))

(think of y by parts... as before...)

Some Functions, etc.
By the weak law of large numbers, if the sample size $n$ is large enough, we can approximate

$$Pr(\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} X_i = \mu) \approx 1.$$

For a given sample $X_1, X_2, \ldots, X_n$, the sample mean is

$$\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i.$$

The variance of the sample mean is

$$Var(\bar{X}) = \frac{Var(X)}{n},$$

where $Var(X)$ is the variance of the individual observations.

The central limit theorem states that as $n$ increases, the distribution of the sample mean $\bar{X}$ approaches a normal distribution with mean $\mu$ and variance $\sigma^2/n$.

To compute $Pr(\bar{X} < \mu - k\sigma/\sqrt{n})$, we use the standard normal distribution with mean $0$ and variance $1$.

We can approximate this probability using the cumulative distribution function (CDF) of the standard normal distribution.

$$Pr(\bar{X} < \mu - k\sigma/\sqrt{n}) \approx \Phi\left(-\frac{k\sigma}{\sqrt{n}}\right),$$

where $\Phi(z)$ is the CDF of the standard normal distribution.

The confidence interval for $\mu$ is

$$\bar{X} \pm z_{\alpha/2} \frac{\sigma}{\sqrt{n}},$$

where $z_{\alpha/2}$ is the $100(1 - \alpha/2)$ quantile of the standard normal distribution.

To find $z_{\alpha/2}$ for a given confidence level $1 - \alpha$, we use the inverse CDF of the standard normal distribution.

$$z_{\alpha/2} = \Phi^{-1}(1 - \alpha/2).$$

This gives us

$$\bar{X} \pm z_{\alpha/2} \frac{\sigma}{\sqrt{n}} = \bar{X} \pm \text{Error Term},$$

where $\text{Error Term} = z_{\alpha/2} \frac{\sigma}{\sqrt{n}}$.

The error term represents the uncertainty in estimating $\mu$ from the sample mean $\bar{X}$.
(defun evals (m a)
  (cond (null m) m
        (t (cons (eval (car m) a) (evals (cdr m) a))))

defun evcon (c a)
  (cond (eq c (car a)) (evcon (cdr a) a)
        (t (eval (car a) a))
        (equal (cdr c) (eq (cadr (evcon (cadr c) a) a) a))))

Two Auxiliary Functions
(a b c)

which returns

((((((x a cons (x) λ) f)),
((c q), ((x a cons (x) λ) f)),
λ
eval).

is evaluated as:

(eval (get-user (64) :eval) ((((((x a cons (x) λ) f)),
((c q), (f) λ), λ) eval),
λ).

E.g.: evaluating the resulting expression. E.g.:

value, which is a lambda or table expression and
the atom which is the function name with its
Calls to functions are evaluated by replacing

Evaluation of function calls
Evaluation of Label Expressions

[Diagram of Lambda Calculus expression]

Which returns \(\lambda\).

\[
(((\lambda x) \text{atom} (\text{lambda} (x) \text{eval})),
((\lambda x) \text{atom} (\text{lambda} (x) \text{eval})),
((\lambda x) \text{atom} (\text{lambda} (x) \text{eval})))
\]

is evaluated as:

\[
(((\lambda x) \text{atom} (\text{lambda} (x) \text{eval})),
((\lambda x) \text{atom} (\text{lambda} (x) \text{eval})),
((\lambda x) \text{atom} (\text{lambda} (x) \text{eval})))
\]

\[\text{eval} \ [9] \ E.9:\]

p.9 [E.9]:

Graham, substituting for the label expression [Graham, an expression with the inner Lambda expression on the environment, and then calling eval on the list of the function name and the function itself, a Lambda expression is evaluated by pushing a]
which returns

\[
(\lambda x \cdot (\text{cons} \ x \ (\text{cdr} \ y))) \circ (\lambda x \cdot (\text{cons} \ x \ (\text{cdr} \ y)))
\]

is evaluated as

\[
(\text{eval} \cdot (\text{cons} \ x \ (\text{cdr} \ y))) \circ (\text{eval} \cdot (\text{cons} \ x \ (\text{cdr} \ y)))
\]

\[
\text{get-user-input} (\text{eval} \cdot (\text{cons} \ x \ (\text{cdr} \ y))) \circ (\text{eval} \cdot (\text{cons} \ x \ (\text{cdr} \ y)))
\]

The argument at, \ldots, an and then evaluating each \( v_i \) to get a list of values \( v_1, \ldots, v_n \). \( \text{eval} \) \( \text{cons} \) \( \text{cdr} \) \( \lambda \)

An expression of the form \((\text{cons} \ x \ (\text{cdr} \ y))\)
Examples
able

gram feature, supporting destructive assignment, and a "pro-
cluded some debugging facilities, and a "pro-
for the IBM 704. In February 1960, this in-
mentation of the LISP programming system
McCarthy's original paper describes an imple-

static layout of the program.
The environment without any concern for the
The language has dynamic scope: label changes

3. Practical numbers

when one has side effects!

2. Sequential execution (which is only useful

I. Side effects (viz., destructive assignment) in

McCarthy's LISP language does not have:

Comments